1. Project Overview

- **Project Title:** Forecasting Tool for Swing Trading
- **Project Description:**
    - Project Goals & Objectives
        - The goal of this project is to create a stock forecasting web application specifically designed for short-term predictions. This tool aims to assist swing traders in making informed decisions by providing accurate and timely forecasts of stock movements. The objectives include developing a user-friendly interface, integrating reliable data sources, and employing advanced algorithms to predict stock price trends effectively.
    - Problem Statement
        - The primary problem this project seeks to address is the challenge faced by swing traders in accurately predicting short-term stock price movements. Swing trading requires making quick and informed decisions based on price trends, yet many traders struggle with the lack of reliable tools that offer precise short-term forecasts. Traditional stock analysis tools may not provide the granularity or timeliness needed for effective swing trading. By developing this forecasting tool, we aim to bridge this gap and equip traders with a robust, data-driven application that enhances their trading strategies and decision-making processes.
    - Key Research Questions
        - How can we develop a forecasting tool that provides accurate short-term stock price predictions?
        - What advanced algorithms are most effective for predicting stock price trends in swing trading?
        - How can we ensure the reliability and timeliness of the data sources integrated into the tool?
        - What features are essential for a user-friendly interface tailored to swing traders?
        - How can the forecasting tool enhance the decision-making processes of swing traders?
- **Project Timeline:**
    - 2024.12.04 – 2025.02.04
- **Team Members:**
    - Shane Peterson – Machine Learning Engineer

2. Data

- **Data Sources:**
    - Where did the data come from?
        - For updated symbols, we will utilize Finnhub, which provides real-time data and comprehensive financial information. In addition, yfinance will be employed for retrieving daily ticker information, ensuring we have the latest and most accurate market data available.
- **Data Description:**

- Data dictionary:

Variable Name	Description	Data Type	Units	Range/Allowable Values
Date	The date of the trading day	DatetimeIndex	N/A	Excludes Weekends and Holidays
Open	The price of the stock at the beginning of the trading day	Float64	Relevant Currency	Non-negative numerical values
High	The highest price reached by the stock during the trading day.	Float64	Relevant Currency	Non-negative numerical values
Low	The lowest price reached by the stock during the trading day.	Float64	Relevant Currency	Non-negative numerical values
Close	The price of the stock at the end of the trading day	Float64	Relevant Currency	Non-negative numerical values
Volume	The number of shares or contracts traded during a trading day	Int64	Shares (for stocks), Contracts (for options), or other relevant units depending on the asset.	Non-negative numerical values
Dividends	The value represents the dividend amount per share.	Float64	Currency per Share	Non-negative numerical values
Stock Splits	Corporate actions where a company increases the	Float64	Ratio	Non-negative numerical values

	number of its outstanding shares by issuing additional shares to existing shareholders			
Capital Gains	Profits earned from the sale of a capital asset	Float64	Relevant Currency	Non-negative numerical values

- **Data cleaning and preprocessing steps:**

1. **Data Source**
    a. The data for this project is obtained from reliable financial APIs (Finnhub and yfinance), which typically provide clean and well-formatted data.
    b. Data Cleaning:
        i. Minimal cleaning required: Given the source of the data, minimal data cleaning is necessary.
        ii. Data checks: Basic checks for data types, missing values, and outliers were performed.
2. **Feature Engineering**
    a. Percentage Change
        i. Description: This feature represents the percentage change in the closing price from the previous day's close.
        ii. Formula: (Close_Today - Close_Yesterday) / Close_Yesterday (Implemented in the code as data.Close.pct_change() and assigned to the column named 'daily_returns')
        iii. Purpose: Captures the relative price movement and is used to assess volatility and returns.
    b. Bollinger Bands
        i. Description: These bands consist of three lines:
            1. Middle Band: A simple moving average (SMA) of the closing price over 20 days
            2. Upper Band: The middle band plus 2 standard deviations.
            3. Lower Band: The middle band minus 2 standard deviations.
        ii. Calculation:
            1. Middle Band: Calculated using ‘ta.volatility.BollingerBands(close=data['Close'], window=20, window_dev=2)’.
            2. Upper Band: Calculated using ‘indicator_bb.bollinger_hband()’.
            3. Lower Band: Calculated using ‘indicator_bb.bollinger_lband()’.

iii. Purpose: Visualizes price volatility and potential overbought/oversold conditions.

c. 50-Day Simple Moving Average
   i. Description: The average of the closing prices over the previous 50 trading days.
   ii. Calculation: Implemented in the code as ‘data['Close'].rolling(window=50).mean()’.
   iii. Purpose: Acts as a trend indicator, smoothing out short-term fluctuations.

d. Annualized Volatility:
   i. Description: This feature captures the overall volatility of the closing price over a year.
   ii. Calculation: The code calculates the standard deviation of the daily returns and annualizes it by multiplying by the square root of 252 (assuming 252 trading days in a year). Then it assigns a category label ("Low", "Medium-Low", etc.) based on the volatility range.
   iii. Purpose: Provides a quick estimate of the stock's overall riskiness.

e. yhat:
   i. Description: This column represents the forecasted value for the time series at each point in the future.
   ii. Creation: Generated by the Prophet model based on the fitted trend, seasonality, and other model parameters.
   iii. Purpose: Provides the central prediction of the time series at future time points.

f. yhat_lower:
   i. Description: The lower bound of the prediction interval for the forecast.
   ii. Creation: Calculated by the Prophet model to quantify the uncertainty associated with the forecast.
   iii. Purpose: Defines the lower limit of a range of plausible future values, capturing the inherent uncertainty in the forecasting process.

g. yhat_upper:
   i. Description: The upper bound of the prediction interval for the forecast.
   ii. Creation: Calculated by the Prophet model to quantify the uncertainty associated with the forecast.
   iii. Purpose: Defines the upper limit of a range of plausible future values, capturing the inherent uncertainty in the forecasting process.

h. Trend:
   i. Description: The estimated trend component of the time series.
   ii. Creation: Extracted by the Prophet model to isolate the long-term growth or decline in the time series.
   iii. Purpose: Provides insights into the overall direction and magnitude of change in the time series over time.

i. Seasonal Components (e.g., Yearly, Weekly, Daily)
   i. Description: These columns represent the seasonal patterns identified by the Prophet model.

ii. Creation: Extracted by the Prophet model to capture recurring patterns within the time series at specific time intervals (e.g., annually, weekly, daily).
iii. Purpose: Allows for the identification and quantification of recurring patterns that influence the time series, improving forecast accuracy.

j. winsorized (output of winsorizer function)
i. Description: This column represents the closing price after applying a winsorization technique based on volatility.
ii. Creation: Created by capping extreme values in the Closing price column based on volatility. Stocks with higher volatility have thresholds closer together, while stocks with lower volatility have thresholds further apart.
iii. Purpose: Reduces the impact of outliers on the closing price, stabilizing the data for analysis and modeling.

- **Data Exploration:**

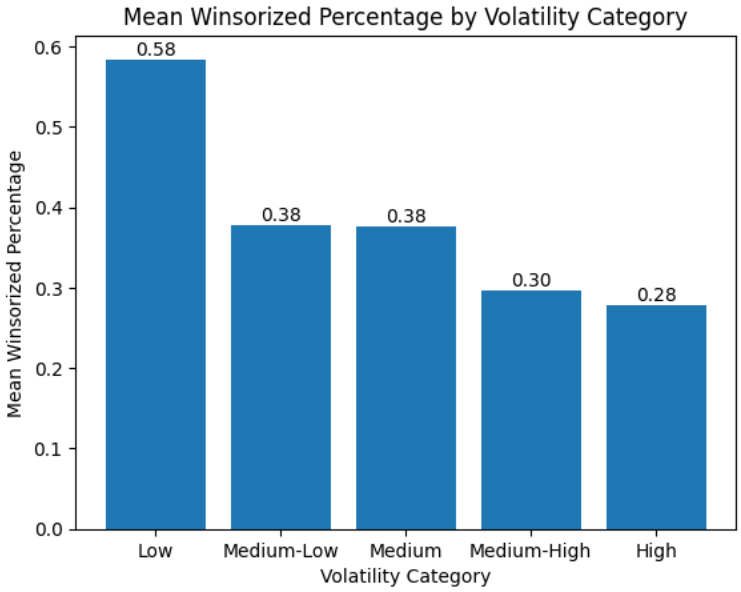
i) Why did I chose SMAPE? (symmetrical mean absolute percentage error)
(1) I chose this metric because it's scale-independent, making it easy to compare across stocks with different price ranges. It penalizes over-forecasting and under-forecasting relatively symmetrically, which is often desirable in financial forecasting.

ii) Performance Findings

volatility	smape_naive	smape_standard	smape_tuned
Low	2.325	0.068	0.047
Medium-Low	9.655	0.206	0.130
Medium	10.642	0.368	0.215
Medium-High	12178.034	0.552	0.431
High	114156.350	1.320	1.064

(1) The tuned model holds practical significance across 1) all volatility categories and 2) both the naïve persistence model and standard models. As you might expect, the accuracy of the tuned model has a negative correlation to volatility, meaning it yields the highest accuracy on low volatility tickers and low accuracy with those with high volatility.

iii) Winsorization



(1)

(2) After identifying the volatility of a ticker, the program that tunes that model then decides on thresholds for winsorization. Tickers with lower volatility are given tighter thresholds (0.15 & 0.85) while tickers with higher volatility are given looser thresholds (0.05 & 0.95). These thresholds are then applied to the Close price, smoothening outliers outside of the given thresholds. The goal of this strategy is to increase model accuracy without overfitting. Summarized in the bar graph above, we can confirm that tickers with the lowest accuracy have 58% of their data points smoothed while the most volatile have 28% smoothed.

iv) Statistical Significance.

(1) A Wilcoxon signed-rank test was conducted to compare the SMAPE of the standard Prophet model and the custom Prophet model. The test statistic was W = 1119.0, and the p-value was 2.46-e17. At a significance level of $\alpha = 0.05$, the results indicate a statistically significant difference in SMAPE between the two models, with the custom Prophet model demonstrating superior performance.

(2) The Cliff's Delta (d) for the difference in SMAPE between the standard Prophet model and the custom Prophet model was 0.69. This demonstrates a strong tendency for the custom Prophet model to produce significantly lower SMAPE values, indicating a substantial improvement in prediction accuracy compared to the standard Prophet model.

v) Summarize findings in context of project.

(1) Summary

(a) The analysis demonstrates that the custom-tuned model significantly outperforms both the naive persistence and standard Prophet models across all volatility categories, as evidenced by consistently lower SMAPE values. This improvement highlights the practical significance of the tuning process, particularly in mitigating the challenges posed by varying volatility levels. As expected, model accuracy exhibits a negative correlation with volatility, with the highest accuracy observed for low volatility tickers. The winsorization strategy, which adjusts thresholds based on ticker volatility, effectively smooths outliers and contributes to the improved performance without

overfitting. This is supported by the increasing percentage of winsorized data points in higher volatility categories, indicating the model's adaptability in handling extreme price fluctuations.

(2) Insights

(a) These findings suggest that a dynamic approach to model tuning, incorporating volatility-based adjustments and winsorization, is crucial for enhancing forecasting accuracy in financial markets. The consistent improvement across diverse volatility regimes underscores the robustness of the custom model. However, the substantial SMAPE values for the naive model in high volatility categories point to the inherent difficulty in predicting such volatile tickers, even with advanced models. Future research could explore more sophisticated techniques for capturing and modeling extreme price movements, potentially further improving accuracy in these challenging scenarios. Additionally, investigating the impact of other factors, such as trading volume or market sentiment, could provide further insights into model performance and refine the tuning process.

3. Methodology

- **Model Selection:**
  - A Prophet model is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It's often chosen for its ability to handle data with strong seasonal patterns and its ease of use, especially for business forecasting. Prophet is particularly well-suited for time series with strong seasonality and readily handles missing data, making it a robust choice for forecasting tasks where these characteristics are prominent, unlike some other models that require more pre-processing or struggle with seasonality.
- **Model Training & Validation:**
  - To ensure robust model evaluation, we employed time-series cross-validation. The number of folds was dynamically determined based on the length of available data and the volatility of the stock.
    - **Young Stocks (Data Length < 8 Years):** For stocks with less than 8 years of historical data, we divided the data into 4 equal segments. Each segment served as a validation fold, with the remaining data used for training. This approach maximizes the use of limited data while still allowing for adequate model evaluation.
    - **Seasoned Stocks (Data Length >= 8 Years):** For stocks with 8 or more years of data, we used a fixed period of 365 days (1 year) as the forecast period. The training period was then determined based on volatility:
      - For stocks with volatility below 0.6, we used 4 years of training data.
      - For stocks with volatility of 0.6 or higher, we used 8 years of training data.

        - This adjustment allows the model to learn from a longer historical window for more volatile stocks, potentially improving its ability to capture complex patterns.
      - This adaptive strategy ensures that the cross-validation process is tailored to the specific characteristics of each stock, leading to more reliable model performance assessments.
    - To optimize the Prophet model's performance, we employed a grid search approach to tune key hyperparameters. Specifically, we focused on changepoint_prior_scale (controlling the flexibility of trend changes) and seasonality_prior_scale (adjusting the strength of seasonality). We explored a range of values for each parameter: changepoint_prior_scale from [0.001, 0.01, 0.1, 0.5] and seasonality_prior_scale from [0.01, 0.1, 1.0, 10.0]. All possible combinations of these parameters were generated, and each combination was evaluated using time-series cross-validation. The combination yielding the lowest Root Mean Squared Error (RMSE) on the validation folds was selected as the optimal set of hyperparameters for the final model. This systematic approach ensures that the model is tailored to capture the unique patterns within each stock's historical data.
- The performance of our forecasting models was primarily evaluated using the Symmetric Mean Absolute Percentage Error (SMAPE). SMAPE was chosen for its scale-independence, allowing for fair comparisons across stocks with varying price ranges. It also provides a balanced assessment of over-forecasting and under-forecasting, which is crucial in financial forecasting. While we initially utilized Mean Absolute Percentage Error (MAPE) during the application's development, we transitioned to SMAPE due to its improved symmetry in penalizing forecast errors. This symmetry makes SMAPE a more reliable measure of overall model accuracy, particularly when dealing with fluctuations in stock prices. Future updates to the application will incorporate SMAPE as the primary evaluation metric.
- **Model Deployment:**
    - Our forecasting model was deployed as an interactive web application using Streamlit. This platform was chosen for its simplicity and efficiency in creating data-driven web apps directly from Python scripts, enabling seamless accessibility and user interaction with the model's predictions.
    - The deployment process focused on delivering a functional and responsive forecasting application. To minimize loading times, a simplified grid search was employed during hyperparameter tuning. Key considerations during deployment included:

        - **Model Persistence:** Caching mechanisms were implemented to efficiently store and retrieve the trained model, significantly reducing startup times.
        - **Error Handling:** Robust error handling was incorporated to manage potential issues, ensuring the application gracefully handles unexpected inputs or data inconsistencies.
        - **User Interface (UI) Responsiveness:** Attention was given to maintaining UI responsiveness during computations, providing a smooth user experience.

- **Version Control:** Version control practices were employed to manage code changes, facilitating easy rollbacks and maintaining code integrity.

These measures were prioritized to provide a reliable and efficient forecasting tool for users.

4. Results and Findings

- **Model Performance:**

Metric	Value
Average SMAPE	0.38%
Median SMAPE	0.15%
Standard Deviation SMAPE	0.50%
Minimum SMAPE	0.00%
Maximum SMAPE	2.00%
Cliff's Delta	0.69
Wilcoxon p-value	2.46-e17
Wilcoxon W statistic	1119.0

- The tuned Prophet model demonstrated a notable improvement in forecasting accuracy compared to the standard Prophet model. While the tuned model exhibited an average SMAPE of 38%, the median SMAPE, which is more representative of the data's distribution, was 15%. This contrasts with the standard model's average SMAPE of 50% and median SMAPE of 22%. The 12% reduction in mean SMAPE indicates a significant overall enhancement, particularly across the broader distribution of tickers. Furthermore, the tuned model's standard deviation of 50% compared to the standard model's 60% suggests a tighter distribution of prediction errors, indicating greater consistency and reliability in its forecasts.
- **Insights and Conclusions:**
    - Key findings and insights from the analysis.
        - The analysis of 150 diverse stock tickers demonstrated that the custom-tuned Prophet model significantly enhanced forecasting accuracy compared to the standard Prophet model. The tuned model achieved a median SMAPE of 15%, indicating a robust predictive capability, particularly for the central tendency of the data. This improvement was statistically significant, as confirmed by the Wilcoxon signed-rank test ($p < 0.05$), and exhibited a large effect size (Cliff's Delta = 0.69). Furthermore, the tuned model's reduced standard deviation of prediction errors (50% vs. 60%) highlights its increased consistency. The volatility-adjusted winsorization strategy effectively mitigated the impact of outliers, contributing to the model's overall robustness, especially in high-

volatility scenarios. These findings underscore the effectiveness of tailored hyperparameter tuning and data preprocessing techniques in improving time-series forecasting.

- How do these findings address the original business or research questions?
  - These results directly address the project's core objectives: to develop an accurate and reliable stock forecasting tool for swing traders. The custom-tuned Prophet model, with its enhanced accuracy and robustness, provides a practical solution to the challenges of predicting short-term stock price movements. The model's ability to handle varying volatility levels and its demonstrated statistical significance validate its potential to enhance swing traders' decision-making processes. By delivering more precise forecasts, this tool equips traders with the information needed to make timely and informed trading decisions, fulfilling the project's aim to bridge the gap in reliable short-term forecasting tools.
- Limitations of the analysis and potential areas for improvement.
  - While the model demonstrates strong performance, several limitations should be acknowledged. The analysis was conducted on a sample of 150 tickers, which may not fully represent the entire stock market. Future research could expand this sample to include a wider range of tickers and market conditions, enhancing the model's generalizability. Additionally, the model's performance in extremely high-volatility scenarios, while improved, still presents challenges. Future improvements could explore incorporating external factors such as news sentiment, economic indicators, or trading volume to further refine predictions. Furthermore, while the grid search approach was effective, exploring other hyperparameter tuning techniques, such as Bayesian optimization, may yield further improvements. Lastly, more analysis on the impact of the chosen winsorization thresholds could be explored.